A Comparison of Robustness Metrics for Scheduling DAGs on Heterogeneous Systems

Louis-Claude Canon, Emmanuel Jeannot

A Comparison of Robustness Metrics for Scheduling DAGs on Heterogeneous Systems

Louis-Claude Canon and Emmanuel Jeannot

LORIA, INRIA, Nancy University, CNRS
Campus Scientique – BP 239
54506 Vandoeuvre-lès-Nancy Cedex, France
{louis-claude.canon,emmanuel.jeannot}@loria.fr

***Abstract*— A schedule is said robust if it is able to absorb some degree of uncertainty in tasks duration while maintaining a stable solution. This intuitive notion of robustness has led to a lot of different interpretations and metrics. However, no comparison of these different metrics have ever been preformed. In this paper, we perform an experimental study of these different metrics and show how they are correlated to each other in the case of task scheduling, with dependencies between tasks.**

I. Introduction

Research in scheduling has gathered a lot of different solutions depending on the pursued objective. For instance, if the objective function to minimize is the *makespan* (the total execution time of the application) different heuristics have been proposed in the literature such as HEFT [17], CPOP [17], hybrid remapper [11], BIL [12], hybrid method [13] or GDL [16]. However, there are a lot of other possible objectives than minimizing the makespan. Among these objectives the *robustness* has recently received a lot of attention [1], [3], [5], [7], [14], [15]. A schedule is said robust if it is able to absorb some degree of uncertainty in the task duration while maintaining a stable solution. Thus, it is important to note that the robustness alone is not a metric but it gives an idea of the stability of the solution with regards to another performance metric such as schedule length, load balance of an application, queue waiting time of batch scheduler, etc. The reason why robustness is becoming an important objective is the recent focus on large systems that can be dynamic and where uncertainty in terms of workload or resource usage can be very important. Moreover, a brief look at the literature shows that despite the fact that robustness is a very intuitive notion there is no consensus on a single metric. Conversely, almost each paper uses its own metric depending on the studied problem and the general context of the work. Furthermore, there does not exist a comparison between these different metrics, hence it is not possible to decide which metric to use when designing a heuristic.

In this paper we focus on comparing different metrics of robustness in the context of scheduling task graph on heterogeneous systems: we model an application as a set of tasks having precedence constraints and a task as a set of statements. The performance metric we use is the makespan (the completion time of the application) and therefore, we look at the robustness of the makespan when tasks may have variations in their duration. Moreover, we try to see to which extend optimizing the makespan can help in optimizing the robustness. In other words, we try to answer the following question: *are short schedules more robust that long ones?* In this work we also test some makespan-centric scheduling heuristics of the literature (BIL, HEFT, Hyb.BMCT) and see on different scenarios how they perform in terms of robustness.

Therefore, the contribution of this paper is the following: we provide a comprehensive study of different robustness metrics in the case of task graph scheduling. We study how they are correlated to each other and whether robustness and makespan are conflicting objectives or not. Finally, we compare the robustness of three different makespan-centric scheduling heuristics.

The remaining of the paper is organized as follows. In Section II we present the problem and the notations used in this paper. Several works dealing with robustness are detailed in Section III. The robustness metrics we use are described in Section IV. In Section V we present the experimental setup we used for testing and comparing the different metrics. Results are shown in Section VI and discussed in Section VII. Finally, conclusion and future works are given in Section VIII

II. Models

We model the parallel application by a directed acyclic graph (DAG) $G = (V, E, C)$, where V is a set of nodes that represent tasks and E is a set of edges that represent dependencies between tasks (often due to communications). C is the set of communication volume between tasks. The target platform is composed of a set of heterogeneous resources each having different capacities in terms of network speed. When there is no uncertainty we use two matrices to model communication speed: $T = (\tau_{i,j})_{1 \leq i \leq m, 1 \leq j \leq m}$ and $L = (l_{i,j})_{1 \leq i \leq m, 1 \leq j \leq m}$, where m is the number of machines. $\tau_{i,j}$ is the time to send one data element from processor i to processor j and $l_{i,j}$ is the network latency from processor i to processor j. To model the fact that communications are way faster between two tasks mapped on the same processor and thus negligible, we put $\forall i \in [1, m],\ \tau_{i,i} = l_{i,i} = 0$. Hence, if task 1 is mapped to processor i and task 2 is mapped to processor j then the communication time between these two tasks will be: $l_{i,j} + c_{1,2} \times \tau_{i,j}$, where $c_{1,2} \in C$ is the communication volume between task 1 and task 2. As we

want to deal with the case where communication times can vary from one execution of the schedule to another execution we enhance the previous model with random variables. In this case, each $l_{i,j}$ and $\tau_{i,j}$ are drawn randomly from a random variable (which can be of any type). We use two parameters: the *minimum* value of this random variable and the *uncertainty level* (UL). The UL is such that the minimum value times the UL gives the maximum value. The idea behind this ratio is that the larger the task duration, the larger the possible values of different execution times are. Also, the larger the UL, the larger the possible values of the random variable are. Based on these two values, it is possible to compute the probability density of the corresponding random variable.

To determine the computation time of a task we use the *unrelated model*. This means that for each task, the minimum duration on each processor is given by a matrix of n rows and m columns, where n is the number of tasks. We also consider that the computation times may vary and thus, we use the same UL as when we compute the communication time to the maximum task duration and determine the random variable that describe the task duration in case of uncertainty. Although this model has some limitations, it is sufficient to handle many different heterogeneity and uncertainty cases.

A schedule is the assignment of the tasks to the processors with a start date and an end-date. In this work we consider only *eager schedule* this means that each task, once allocated to a processor starts as soon as possible in the same order that given by the schedule. This means that there is no arbitrary delay (or slack) in the schedule. Note that most of the scheduling heuristics (list, clustering, etc.) produce eager schedule.

We call M the makespan of a *realization*. A realization is computed by instantiating every computation and communication durations according to the random variables. M is then the end-time of the last task to finish for this realization.

Given a schedule, it is possible to have an infinite number of realizations and hence and infinite number of makespans. However some makespans are more likely to occur than others. This is why we introduce the notion of makespan distribution. Given a schedule S we call f_S the makespan probability density function (PDF). With f_S, one can compute the probability that the makespan is within two bounds $[x_1, x_2]$ (noted $(\mathcal{P}(x_1 \leq M \leq x_2))$ and is given by $\int_{x_1}^{x_2} f_S(x)dx$. We will also use the cumulative distribution function (CDF) of the makespan F_S. F_S is the integral of the probability density function f_S. Therefore $F_S(x)$ gives the probability that the makespan of schedule S is lower than x (noted $\mathcal{P}(M \leq x)$).

The probability density of the makespan comes directly from the distribution of the task duration and communication time. Computing numerically the probability density or the CDF of the makespan is computing intensive for task graph with independent task or DAG in which the distributions are independent (an in-tree for instance) but is tractable. In the case of independent distributions, only two cases need to be consider (see [9], [10], for the details). The first case is when a distribution is the ancestor of another distribution. The resulting distribution is computed by adding the two distributions together. The sum of two distributions is computed by doing the convolution of the two probability density distributions and can be calculated numerically using Fast Fourier Transform (FFT). The other case is when two distributions are independent and join to another one. In this case we need to compute the maximum of the two distributions. The maximum of two independent distributions is done by multiplying their CDF. Here again, it can efficiently be calculated by finding the derivative of the probability density and integrating the result.

In the general case, however, a DAG can have a structure such that distributions are not independent. In this case, computing the probability distribution of the makespan is extremely difficult: in the general case it is #P-complete[1] (see [8] for the details). Several authors have proposed solutions to approximate the distribution of the makespan for this case. Among these methods two are of interest for our problem. The Dodin method [6] uses a succession of reductions applied to a given series-parallel graph. This results in a sole node whose random variable is equivalent to the makespan distribution of the complete graph. A mechanism is used to transform any graph into a series-parallel one with some approximation. This is one of the oldest methods and it gives acceptable accuracy. The second method, from Spelde [10], is based on the central limit theorem which states that the sum of random variables tends to be normally distributed. Every random variable is then simplified to its unique mean and standard deviation (the only parameters needed to characterize any normal distribution) and the makespan is calculated without doing any convolution. Thus, this is a fast approximation method although it assumes the independence of the random variable. Refer to [10] for a description and a comparison of these methods. Moreover, the above methods where designed for an unbounded number of processors. In our case, since the number of processors is bounded we have to modify the graph to obtain a distribution of the makespan that corresponds to a given schedule. This is done by adding edges between independent tasks when they are scheduled consecutively on the same processor (such a graph is called the *disjunctive graph*, see [15] for the details).

III. Related Work

How to measure robustness is a subject that has not yet lead to a wide accepted metric. Several works propose different ways to measure this metric. In [1] the authors do a good job in defining how to measure robustness: 1) defining the performance features that need to be robust, 2) identify the parameter that impacts the parameters that impact the robustness are the duration of each task and each communication. Hence, a schedule is said more robust than another one if it requires a greater change of the task duration to exceed some given bounds. The problem of that definition is that it is hard to take into account the fact that some change in task or communication duration are more likely to occur than others. Moreover, computing this metric requires a lot of effort and depends on the studied system.

[1] intuitively a #P problem consists in counting the number of solutions of an NP problem.

In order to simplify the computation of the robustness, [7] proposes to use the Kolmogorov-Smirnov (KS) distance between the CDF of the performance metric under normal operating condition and the CDF of the same performance metric when perturbations occur. The idea is that if the KS distance is large (close to 1) this means that the two distributions are different and thus, that perturbation has a large impact on the behavior of the studied system. However, in many cases, the performance metric under normal operating condition has only one value (think for instance of the arrival time of the train at a station). In this case the distribution is a Dirac function and the CDF is a *step* function. Moreover, if this value is computed using the minimum of each intermediate event, the KS distance is always 1 whatever the way you organize the system. This means that this metric is not well adapted to the case where the performance metric has only one possible value, which is the case for the scheduling problem studied here.

In [14] a subset of the authors of [1] proposes a new metric called the probabilistic metric. It is defined as the probability that the performance metric is confined within a given interval. They evaluate this metric against the robustness radius (called the deterministic robustness in the paper) and show that the probabilistic metric is preferable to the deterministic metric in the case of independent tasks scheduling.

Other definitions of the robustness are available in the literature. In [3] Bölöni and Marinescu propose to use the slack as a robustness metric. The slack of a task represents a time window within which the task can be delayed without affecting the makespan. The same authors suggest also to use the entropy of the performance metric distribution to compare schedules with the same makespan: given two schedules with the same makespan they conjecture that the one with the smallest entropy is the most robust. In [15] the authors study another definition of slack and show that it is equivalent to the definition given in [3]. They propose two new robustness metrics for the scheduling problem. One is based on the average delay between the expected makespan and different realizations of the schedule under perturbation and the other is the ratio of realization that are late compared to the expected makespan. Moreover the authors show that minimizing the makespan is a contradictory objective with the problem of optimizing the robustness.

This brief look at the literature shows that there is no consensus on a good metric for robustness. This exemplifies the need for a comparison and a systematic study of different metrics in order to determine how these metrics are correlated to each other.

IV. Robustness Metrics

As there is no consensus on a good metric definition, we will compare some metrics proposed in the literature to each other. However, not every metric is easy to implement. In our case we consider the makespan as the performance metric. This means that, for our problem, the robustness we measure is the stability of the makespan whatever the different realizations of the same schedule we can have. Given a task graph and a target environment, we will schedule the tasks and compute the makespan distribution. Let f be the PDF of the makespan of the schedule, F the CDF of the makespan of the same schedule and $E(M)$ the expected makespan (the average value of the makespan). Based on these definitions we define the following robustness metrics.

- **Makespan standard deviation**. Intuitively the standard deviation of the makespan distribution tells how narrow this distribution is. The narrower the distribution, the smaller the standard deviation is. This metric is related to the robustness because when you are given two schedules the one for which the standard deviation is the smaller is the one for which realizations are more likely to have a makespan close to the average value. Mathematically we have:
$$\sigma_M = \sqrt{E(M^2) - E(M)^2}$$
- **Makespan differential entropy**. The differential entropy of a distribution measures the uncertainty of that distribution. If there is less uncertainty there is more chance than two realizations give a close result and hence that the schedule is robust.
$$h(M) = \int_{-\infty}^{+\infty} f(x) \log f(x) dx$$
- **Average Slack**. The slack gives the sum of spare time in the schedule [3]. It is intuitively related to the robustness of the makespan as a schedule with a large slack is able to absorb a lot of uncertainty. For a deterministic schedule the slack is defined as
$$S = \sum_{i \in V} M - \text{Bl}(i) - \text{Tl}(i)$$
Where M is the makespan, $\text{Bl}(i)$ is the bottom level of task i (the length of the longest path from i to an exit node including i) and $\text{Tl}(i)$ is the top level of node i (the length of the longest path from an entry node to node i excluding i). In our case we have random variables that define tasks and communications duration. Hence, we compute an approximation of the average slack by taking the average value of the makespan, the task duration and the communication duration.
- **Slack standard deviation**. Each task has its own slack. Some tasks have a very large slack and other a slack of zero. However, as shown in [15] only task with non-zero slack can absorb uncertainty without delaying the makespan. Hence, it is better to have as many tasks as possible with a slack close to the average, this means that the standard deviation of all the slacks needs to be as small as possible. In order to compute the standard deviation of the slack we use the average slack S as defined above and the slack of every node $i \in V$: $s_i = M - \text{Bl}(i) - \text{Tl}(i)$. Then, we have the standard

deviation of the slack being:

$$\sigma_S = \sqrt{\sum_{i \in V} (s_i - S)^2}$$

- **Average lateness**. A schedule is said late if its makespan exceeds the average makespan. The average lateness as defined in [15] is the average of the difference between the makespan of the late realization and the average makespan. If this metric is large this means that the makespan tends to be far from the average and then that the robustness is low. It is defined as:

$$L = E(M') - E(M)$$

where M' is the random variable describing the realizations that have a makespan larger than E(M).

- **Probabilistic metric**. This metric has been defined in [14] and gives the probability that the makespan is within two bounds. If this probability is high, this means that the makespan of a given realization is likely to be close to the average makespan and hence that the robustness is high. We propose two variants of this metric. An absolute probabilistic metric that measures the probability of the makespan to be within $[E(M) - \delta, E(M) + \delta]$ where $E(M)$ is the average makespan and δ a positive constant given by the user. We also propose the relative metric that measure the probability of the makespan to be within $[E(M) \times \frac{1}{\gamma}, E(M) \times \gamma]$, where γ is a real number greater than 1. Formally, The absolute probabilistic metric is defined as:

$$A(\delta) \quad = \mathcal{P}(E(M) - \delta \leq m \leq E(M) + \delta)$$

and the relative probabilistic metric is defined as:

$$R(\gamma) \quad = \mathcal{P}(\tfrac{E(M)}{\gamma} \leq m \leq \gamma E(M))$$

V. Experimental Setup

Since our comparison of these metrics is mostly empirical, great care should be taken to ensure the correctness of our conclusions. Then, considerable attention was given to validate the methodology, which involves the input task graphs, the metrics evaluation and the exploitation of the results.

Several graphs were generated to study the correlations in the general case (with random graph) and in two real-application cases (namely, the Cholesky decomposition and the Gaussian elimination [4]). The generation consists of two phases: obtaining a deterministic graph and transforming it into a stochastic one. For each kind of graph, we vary the number of tasks ($n = 10$, 30, 100 and 1000) and the degree of uncertainty (UL $= 1.01$, 1.1). Additionally, we generated up to 10 different random graphs for each size. The latency was not considered because its influence was negligible on the correlation results. The random generation of DAG requires some parameters, among which the communication-to-computation ratio (CCR $= 0.1$), the coefficient-of-variation ($V_{\text{task}} = V_{\text{mach}} = 0.5$) and the average computation cost of each task of the DAG ($\mu_{\text{task}} = 20$). The idea behind the coefficient-of-variation is to define a ratio between the mean and the standard deviation of each weight in order to have a relative dispersion metric (see [2] for more details). In our case, we apply a Gamma distribution and obtain every deterministic computation and communication weights with these parameters. Finally, concerning the shape of these random graphs, each new node can only connect to the ones at higher level and the out degree is uniformly chosen between one and the sum of all nodes at higher levels. For real-application graphs, only the weight of communications is considered (not the bandwidth) in order to have values with the same order for the processor and the communication times. The computation time of each task on each processor is chosen uniformly in the interval $[\text{minVal}; 2 \times \text{minVal}]$, where minVal is the minimum processing time and is chosen randomly. Once the graphs are generated for the deterministic heterogeneous case, we apply the substitution of fixed values by stochastic ones using random variable as detailed above. We use the Beta distribution and select the parameters in order to have a probability distribution corresponding to our observations and expectations. To this purpose, we need a well-defined nonzero mode (implying $\alpha > 1$) and more small values than large values (meaning we should have a right-skewed probability distribution and thus $\beta > \alpha$). Therefore, we selected $\alpha = 2$ and $\beta = 5$.

The metrics evaluations were performed with a C program using the GSL library for numerical analysis (random generation, FFT, interpolation and smoothing). Moreover, precision and efficiency are guaranteed by the use of some classic numerical technique such as Simpson integration and Overlap-Add methods (for optimizing the convolution). Experimentation shows that sampling each probability density with 64 values was largely sufficient with cubic spline interpolation. On the scheduling side, random schedules are created by repeating iteratively the following three phases: 1) choose randomly a task among the ready ones, 2) assign it to a randomly selected processor and schedule it eagerly, 3) update the list of ready tasks. As stated before, HEFT, BIL and Hyb.BMCT scheduling heuristics were also implemented and validated with simple examples. Their performances on larger graphs are excellent and consistent (a consequence of the low degree of unrelatedness of the task graphs). The evaluation of the makespan distribution (needed for most of the metrics) was realized with Dodin and Spelde methods [10], both gave similar results to the classical algorithm (which assume the independence between random variables when calculating the maximum). The simplest of these methods was used (*i.e.*, assuming independence of the random variables) and its accuracy was measured for the worst cases, revealing that for large graphs the independence assumption does not stand anymore (see Figure 1). Indeed, we used two metrics to evaluate the distance between the CDF of the makespan using the independence assumption and the real CDF of the makespan computed by running 100 000 realizations. The first metric is the Kolmogorov-Smirnov (KS) that measure the maximum distance between the two CDF and the second is

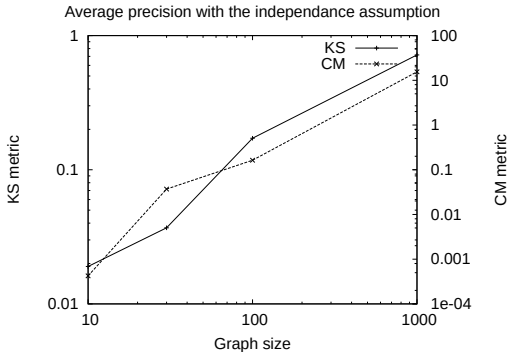

Fig. 1. Average precision of the makespan distribution evaluation with UL = 1.1

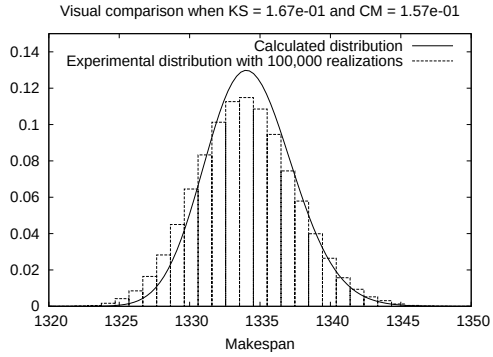

Fig. 2. Visual example of the worst accepted values for KS and CM and the corresponding imprecision

a variant of the Cramér-von-Mises (CM) that measure the distance in terms of area. Hence, although the independence assumption shows some inaccuracy, we have found that a KS value of 0.1 (which is mediocre) does not necessarily imply that the correlation between metrics will be altered (especially if the CM metric is correct, *i.e.* lesser than 0.1). Therefore we have kept graphs having up to 100 nodes (those with 1000 nodes only serving as indications). The Figure 2 reveals that even with poor KS and CM values, our approximation is still close to the experimental realizations. Many metrics calculations are based on the makespan distribution and are thus straightforward to compute and to validate. For the *probabilistic metric*, we have chosen $\delta = 0.1$ and $\gamma = 1.0003$ in order to have values well distributed on the interval $[0; 1]$ (for different ULs, communication costs or processor weights than the one we used here, these values should be adapted). Measuring the slack is quite effortless, since it consists in checking the equality between the bottom level of the first task and the sum of the top level and the bottom level of the last task. The overall program was checked to assure a correct memory usage (including memory access and memory leak).

On the overall we have generated 52 cases with different graphs type, number of nodes, target platform, uncertainty level, etc…For each generated cases, we built 10000 random schedules (2000 for those having $n = 100$) plus one schedule for the 3 heuristics we have implemented (BIL, HEFT, Hyb.BMCT). Even for the smallest graphs, the probability to get the same random schedule twice is not high and these quantities are sufficient for correlation measures. Each metric is then compared to each other visually and with the statistical Pearson correlation coefficient. Even if this correlation measure only indicates the linear relationship between two variables, it is sufficient for slightly curved set of points as shown later. The final result is two matrices, one with the average Pearson coefficients between each metrics, while the other contains their standard deviation of the Pearson coefficients.

VI. Experimental Results

Among all the graphs we have generated, we have selected three relevant ones that are typical of the general behavior (see Figure 3, 4 and 5). On these figures, every 8 metrics are compared to every other ones: leading to a matrix of 64 scattered elements. On the diagonal of the matrix is given the name of each metric. On the lower part we plot the value of each metrics for the random schedules and the schedules given by the 3 tested heuristics. For instance, on Fig. 3, we plot the value of the expected makespan against the entropy of the makespan on the first column and third row (the makespan is plotted on the x-axis and the entropy on the y-axis). For easing the reading of the plot, we inverted three metrics in order to have the optimization of the metrics corresponding to its minimization (hence good results should be plotted in the lower left corner of the corresponding plot). These metrics are the *slack*, because our initial assumption is that a robust schedule has high slack, and the two *probabilistic metrics*, since we want to maximize the probability to be in an interval. The inversion is done by subtracting the measured value to the maximum that was obtained (for the slack) or to 1 (for the probabilistic metrics cases). We did not invert the other metrics because optimizing them consisted already to minimize them (such as the makespan). Additionally, linear regressions were performed on each plot, in order to visualize the correlation. The upper part of the matrix contains the value of the Pearson coefficients associated with each plot corresponding to the metrics. The higher the correlation, the closer to 1 is the absolute value of the Pearson coefficient. The minus sign for correlation means that the metrics are negatively correlated (if one metric increases, the other decreases). For instance we see that, in Fig. 3 the average slack and makespan are negatively correlated by a value of 0.64.

Since the Pearson coefficients show how the metrics are correlated to each other, they are a good way to sum-up our contribution. Hence, we have plotted in Figure 6 the matrix with the Pearson coefficients of 24 different cases (the one with graph of 100 nodes or less). In this figure we have plotted the average value on the upper part of the matrix and the standard deviation on the lower part. We see, for instance, that the average lateness and the absolute probabilistic metric are highly positively correlated (average Pearson coefficient of 0.981) with a very low standard deviation (0.022).

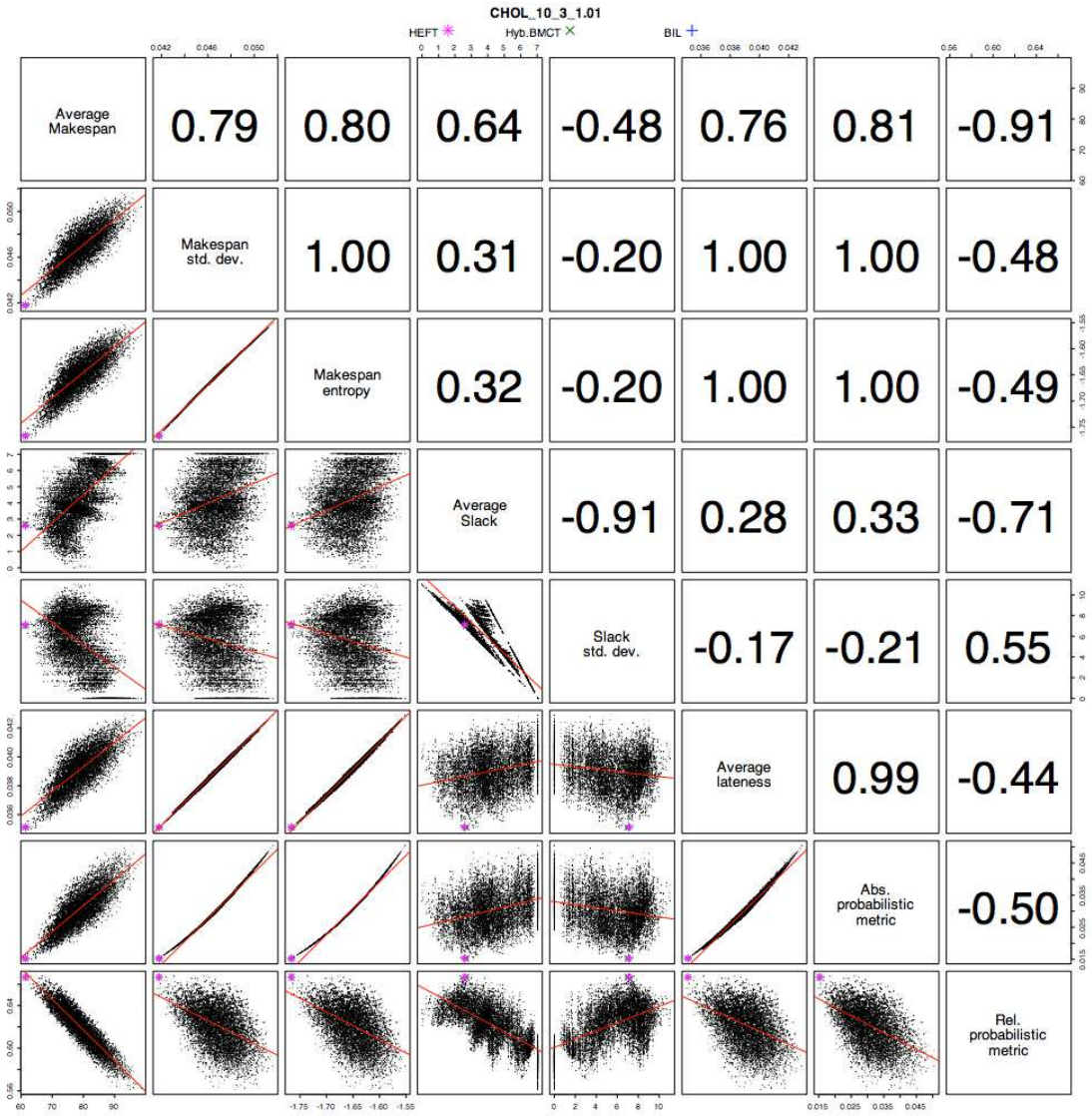

Fig. 3. Metrics correlation for the Cholesky graph of 10 tasks on 3 processors and UL=1.01. Lower part of the matrix: plot for 10000 random schedules and the 3 schedules given by BIL, HEFT and Hyb.BMCT heuristics. Upper part of the matrix: value of the Pearson coefficients for the random schedules.

VII. Discussion

We see immediately the correlation between a number of robustness metrics that are the standard deviation, the differential entropy, the average lateness and the absolute probabilistic metric. Furthermore, we divided the relative probabilistic by the makespan. In this case, we see that it is also correlated to the other ones. This is not plotted on the graphs but the mean Pearson coefficient is 0.998 with a standard deviation of 0.009 when compared to the makespan standard deviation. This relation is common to every graph that was generated, whatever the size, the UL or the type of graph was. The low standard deviation of the Pearson coefficient also indicates that the degree of correlation is almost always the same. Then, these quasi-linear relationships suggest that the probability density shape remains similar for every schedule. An explanation is based on the use of the central limit theorem which states that the sum of random variables having a finite variance (as in our case) will be approximately normally distributed. Indeed, despite the fact that the makespan is obtained by performing a number of operations mixing sum and maximum, the result distribution is really close to a Gaussian (however, in some cases where the last node has many ancestors that are finishing approximately simultaneously, this may not be true). This hypothesis justifies the correlation between these metrics in the case we can apply the central limit theorem. Since, it is a convergence result, we analyze the number of sums needed to satisfy the normal approximation in the worst case. Then, we generated a special distribution (which is constructed with a concatenation of Beta distributions, see Figure 7) and study the accuracy of the approximation. We see on Figure 8 that after only 5 sums with itself, our random variable is almost a Gaussian and that after 10, the difference is negligible. Thus,

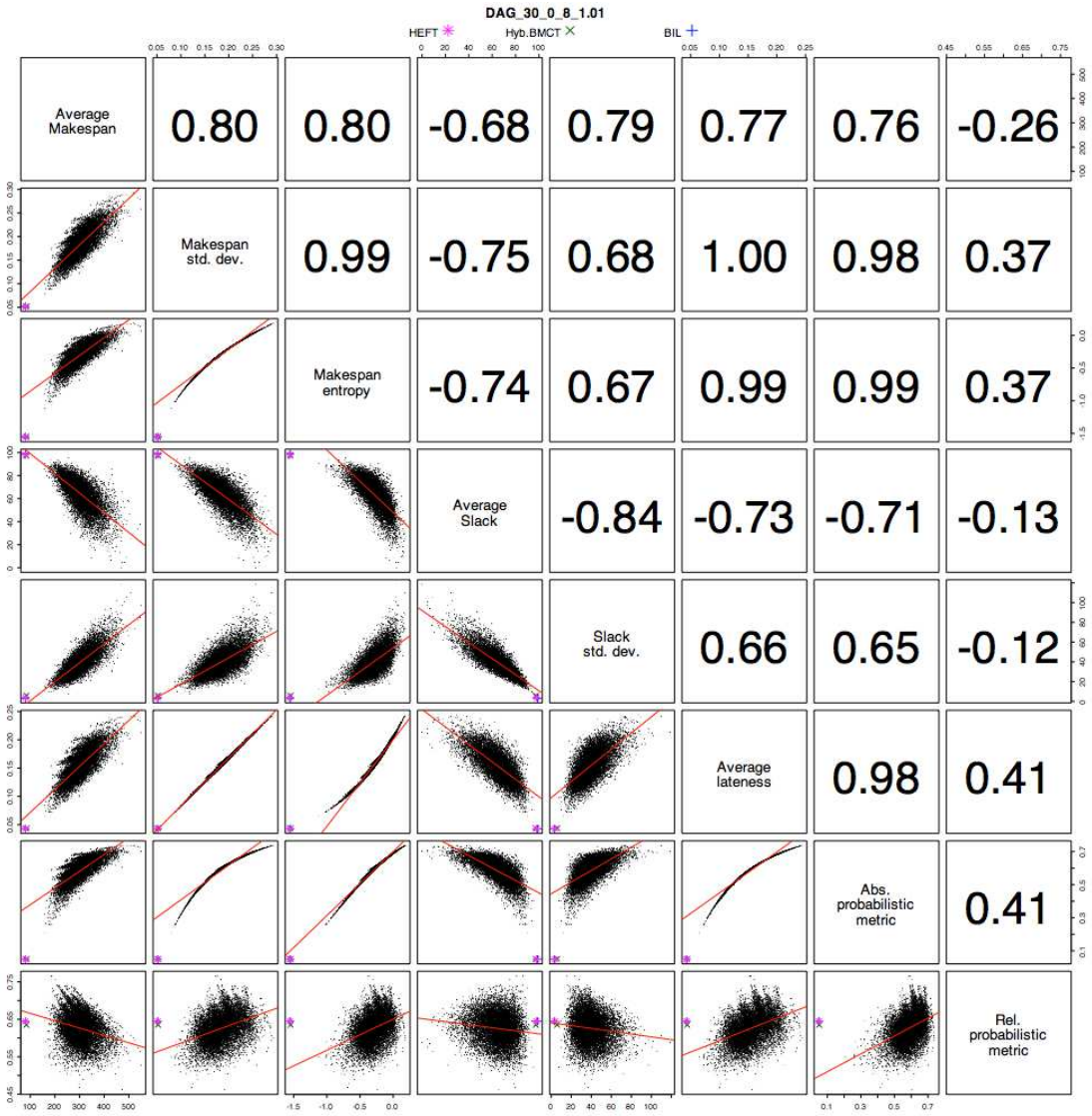

Fig. 4. Metrics correlation for a random graph of 30 tasks on 8 processors and UL=1.01. Lower part of the matrix: plot for 10000 random schedules and the 3 schedules given by BIL, HEFT and Hyb.BMCT heuristics. Upper part of the matrix: value of the Pearson coefficients for the random schedules.

even for small graphs (with only 3 nodes on the critical path) we can simplify the robustness evaluation by calculating only one of the previously mentioned metrics.

A second observation can be made on the relation between the makespan and the slack. If their correlation is not very high and variable (the Pearson coefficient is -0.385 on the average), it is always true that they are conflicting objectives in the sense that optimizing one will produce a poor value for the other metric. Intuitively, a schedule having a good makespan will not have that much unused processor time and a schedule with a lot of slack (or spare time) will not be efficient. The average correlation is due to the existence of schedules with significant makespan and small slack (take the example where all tasks are scheduled sequentially on the same processor).

It is worth noting, too, that the three heuristics (BIL, HEFT and Hyb.BMCT) give always the best makespan and often the best standard deviation (and thus the best of the three other linked metrics). Although the correlation is not excellent and differs to some degree for each graph, there is a noticeable relationship in the general case. To explain this, we have to describe one phenomenon that arises when we are evaluating the makespan probability distribution. The random variable resulting from the sum of two others will have a standard deviation roughly equivalent (equals for the normally distributed random variables, and this is almost our case, see the above argument) to the sum of the two firsts. If we do not considerate the implications of the maximum operator, a direct consequence is that the more tasks on the critical path, the more significant is the standard deviation and hence the final standard deviation will be high. As we modeled the standard deviation to be proportional to the mean of task duration, heuristics producing schedules with low makespan,

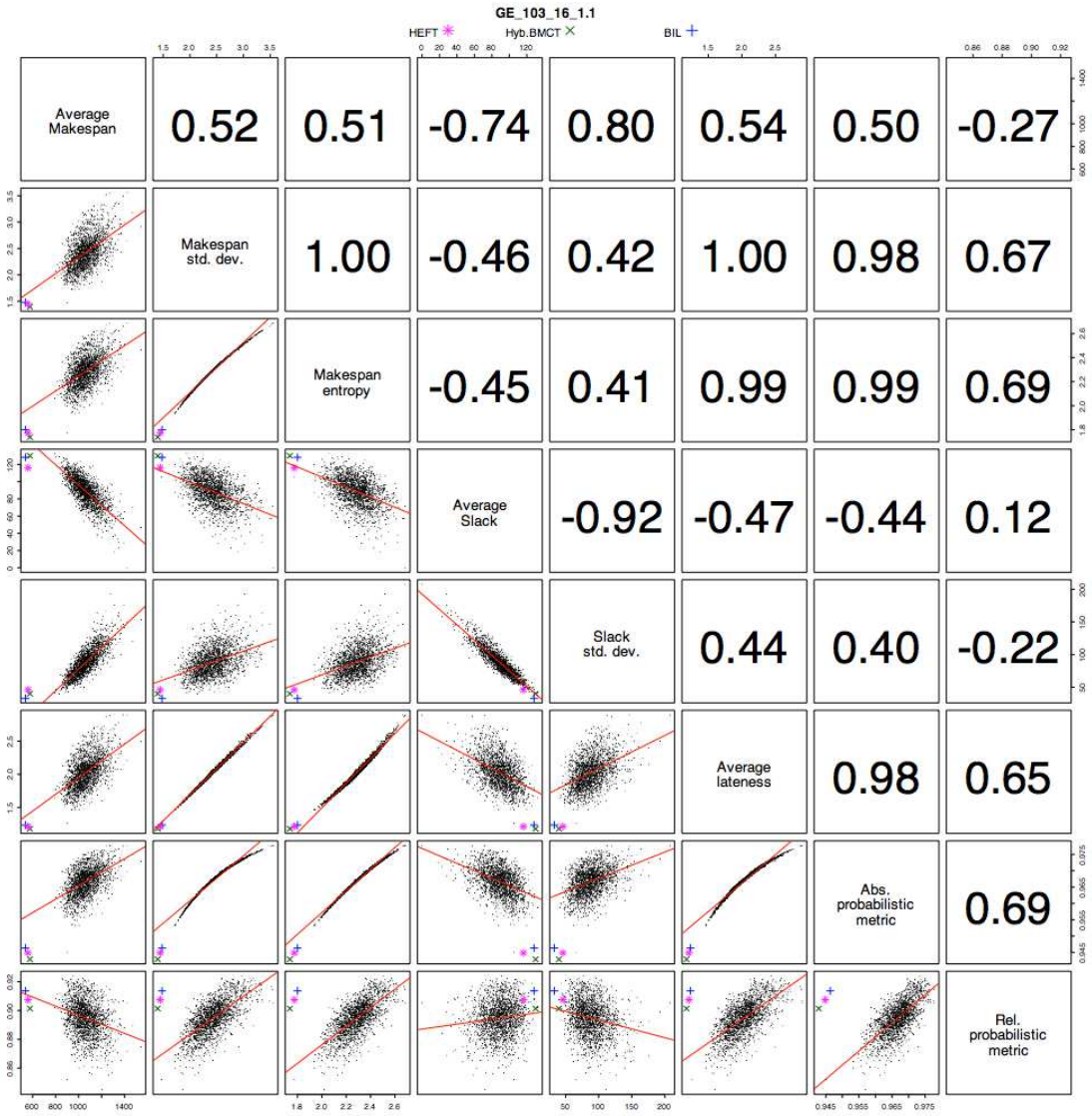

Fig. 5. Metrics correlation for a Gaussian Elimination graph of 103 tasks on 16 processors and UL=1.1. Lower part of the matrix: plot for 2000 random schedules and the 3 schedules given by BIL, HEFT and Hyb.BMCT heuristics. Upper part of the matrix: value of the Pearson coefficients for the random schedules.

hence having less task or shorter tasks on the critical path, will have relatively less standard deviation than schedules with large makespans. The imperfection of the correlation must be due to the existence of the maximum operator.

One surprising result is the low correlation that exists between the slack and the other metrics, and specially the nature of this correlation. Maximizing the slack seems indeed be a conflicting objective with the robustness. This contradicts the intuition that the more slack a schedule will have, the more it will be able to absorb uncertainty. Additionally, some previous work also proposed this metric for robustness. Hence, we present some arguments that confirm this result. The Figure 9 exhibits four examples of schedule for a join task graph of $N+1$ identical tasks having independent and identically distributed (i.i.d.) random variables. Each schedule represents different possibilities with the two objectives being the slack and the standard deviation. The non-robust schedules (according to standard deviation metric) are easy to interpret, which is to say that almost any late task will have a repercussion on the overall makespan. The schedule b) does have a good robustness because only the three tasks on the critical path will have an incidence on the makespan if one of those is late. The schedule a) is more subtle, because it relies on the characteristics of the maximum of two independent random variables being similar. In this case, the resulting mean will be greater than the original means and more importantly, the final standard deviation will be lower than at least the maximum of the two originals. A consequence is that the maximum of an infinite number of i.i.d. random variables is equal to a Dirac (which is completely robust) whose value is the maximum possible value of these random variables. Then, the more tasks we are waiting for, the more we will be sure that one is late,

Pearson coefficients (top: mean, bottom: std. dev.)

Average Makespan	0.767	0.762	-0.385	0.537	0.756	0.734	-0.467
0.107	Makespan std. dev.	0.996	-0.460	0.480	0.999	0.982	0.148
0.109	0.002	Makespan entropy	-0.458	0.476	0.994	0.990	0.154
0.407	0.301	0.299	Average Slack	-0.873	-0.461	-0.444	-0.134
0.373	0.256	0.254	0.028	Slack std. dev.	0.480	0.456	-0.084
0.098	0.001	0.002	0.291	0.248	Average lateness	0.981	0.165
0.104	0.021	0.029	0.299	0.256	0.022	Abs. probabilistic metric	0.184
0.245	0.384	0.386	0.252	0.238	0.375	0.380	Rel. probabilistic metric

Fig. 6. Average (top) and standard deviation (bottom) of the Pearson coefficients for 24 different experiments

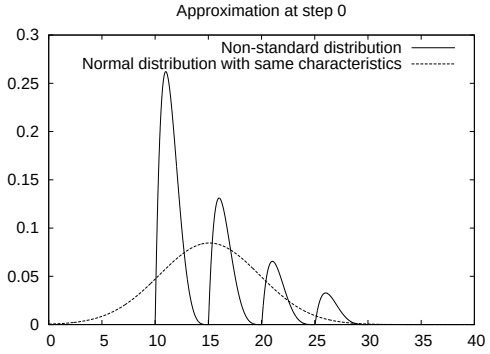

Fig. 7. Two distributions (a special and a normal) having the same mean and standard deviation

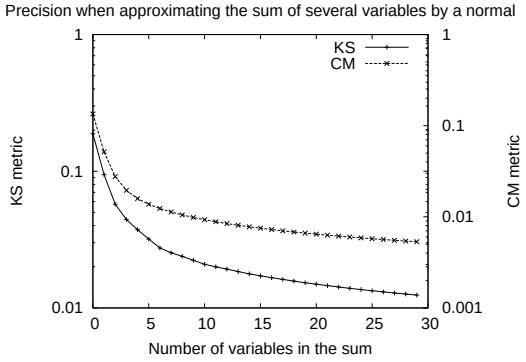

Fig. 8. Precision when approximating the special distribution n-times by a normal distribution after n-sums with itself

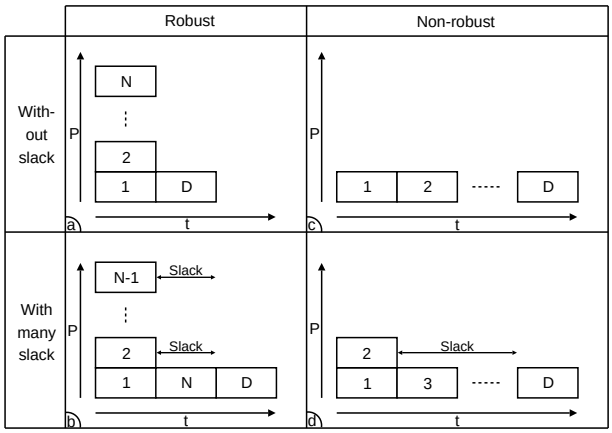

Fig. 9. Four schedules on P processors for different robustness and slackness, considering i.i.d. random variables and a join graph with $N + 1$ tasks

and the more the schedule will be robust because we will have more certainty on the expected maximum. With these four examples, we see that the slack is not necessarily related to the robustness. Moreover we see, as we already explain why, that the slack and the makespan are conflicting objectives and schedules with good makespan are often more robust. These explanations are consistent with the measures showing that slack and robustness are antagonist metrics (also showed in [15]).

The motivation behind the standard deviation of the slack metric comes from the fact that despite following a seemingly pertinent intuition, the sum of all slacks is unable to take into account every situation. Its standard deviation denotes a correct distribution of every slack (a kind of robustness measure on a sub-metric). However, it should be remarked that it does not use the absolute value of the slack and thus a schedule with 0 slack will have a 0 standard deviation. Its correlation with the slack confirms that low values only signify low slack. Then, regarding the above discussion, even though experiments show that it constitutes a slightly better metric than does the slack alone, it does not add real significance to the study.

It was showed in [15] that the slack was related to two robustness metrics (R1, the average lateness and R2, the ratio of late schedule). However, it is conjectured that these metrics were calculated in a way that was favorable to the slack metric. Indeed, the base makespan to which was compared the makespan of realizations was obtained by simplifying each random variable to its means (which is an approximation due to the convexity of the expected value operator for the maximum of random variables). Thus R1 and R2 actually measure the degree of this approximation which is lowered when maximums are performed with random variables having very different means. This case has more chance to happen when there is some slack. Furthermore, and this may be the main reason, the study was restricted to schedule with better makespan than the one given with the HEFT heuristic which reduces the cases of generality of the results. These hypotheses would need to be deeply examined.

A last point deserving our attention is the consequence of the maximum operator. It is stated that the maximum of two i.i.d. random variables is more robust. In our case, the random variables are not independent but the dependence depicted by the task graph does not contradict this assertion. Therefore, it would imply that a way to improve robustness is to increase the similarity of the random variables on which we are making the maximum (then, equilibrating the finish time of the ancestor of every node).

VIII. Conclusion

Robustness is an objective that has led to a lot of different metrics. However, there is no consensus on a wide-accepted metric. Our empirical study was intended to determine the relationship between a comprehensive set of robustness metrics presented in the literature in the case of task graph scheduling with precedence constraints. The first conclusion we can draw is that several of them are equivalent mostly due to the implication of the central limit theorem. Consequently, the simplest of these metrics, certainly the standard deviation, is sufficient in most real cases and denotes the absolute dispersion of the makespan, its lateness, etc. (which are all related). On a more pragmatic aspect, we noticed that the makespan is almost an efficient criteria for the robustness since HEFT, BIL and Hyb.BMCT gave good results. However, since the correlation between these two criterions is not perfect and especially if we do not take a constant UL for a given graph (which will break the equivalence between task duration mean and standard deviation), we believe that the makespan could be a misleading criteria. The last important point is the unsuitability of the slack in our uncertainty model. We presented some arguments to justify this observation and hypothesized that it might be a better metric if the lateness is modeled otherwise (e.g., by constant duration and a probability to be late) or with variable UL or different probability densities.

Future works include:

- Extending the validity of the results to larger graphs with greater and variable UL and with non-standard probability distributions (with some oscillations, for example). Contrarily to what is stated in [9] which establish the validity of the independence assumption when evaluating the makespan distribution, we see that it is clearly not the case for large graphs or/and with bigger ULs.
- Studying the correlation in the extreme cases (near the Pareto front). Our results are indeed obtained with random schedules which only give an indication of correlation between the metrics. However, at some point (for low makespan schedules) there could be some trade-off to find.
- Validating the proposed model and the results obtained from simulations by realizing experimentations on real heterogeneous platform with concrete applications.
- Finding an efficient heuristic similar to classic list heuristic based on the standard deviation of every tasks duration rather than their mean or minimal value. This heuristic should be able to produce good and robust schedules.

IX. Acknowledgement

We would like to thank Frédéric Suter for carefully reading this articles and providing helpful comments on its content.

References

[1] Shoukat Ali, Anthony A. Maciejewski, Howard Jay Siegel, and Jong-Kook Kim. Measuring the Robustness of a resource Allocation. *IEEE Transaction on Parallel and Distributed Systems*, 15(7):630–641, July 2004.

[2] Shoukat Ali, Howard Jay Siegel, Muthucumaru Maheswaran, Debra Hensgen, and Sahra Ali. Representing Task and Machine Heterogeneities for Heterogeneous Computing Systems. *Tamkang Journal of Science and Engineering,* Special 50th Anniversary Issue, 3(3):195–207, November 2000.

[3] Ladislau Bölöni and Dan C. Marinesco. Robust scheduling of metaprograms. *Journal of Scheduling*, 5(5):395–412, September 2002.

[4] Michel Cosnard, Mounir Marrakchi, Yves Robert, and Denis Trystram. Parallel Gaussian elimination on a MIMD computer. *Parallel Computing*, 6:275–296, 1988.

[5] Andrew J. Davenport, Christophe Gefflot, and J. Christopher Beck. Slack-based Techniques for Robust Schedules. In *Proceedings of the Sixth European Conference on Planning (ECP-2001)*, Toledo, Spain, September 2001.

[6] Bajis Dodin. Bounding the project completion time distribution in PERT networks. *Operations Research*, 33(4):862–881, 1985.

[7] Darin England, Jon Weissman, and Jayashree Sadagopan. A New Metric for Robustness with Application to Job Scheduling. In *14th IEEE International Symposium on High Performance Distributed Computing (HPDC-14)*, pages 135–143, July 2005.

[8] Jane N. Hagstrom. Computational complexity of PERT problems. *Networks*, 18(2):139–147, 1998.

[9] Yan Alexander Li and John K. Antonio. Estimating the Execution Time Distribution for a Task Graph in a Heterogeneous Computing System. In *6th IEEE Heterogeneous Computing Workshop (HCW'97)*, pages 172–184, Geneva, Switzerland, April 1997.

[10] Arfst Ludwig, Rolf H. Mohring, and Frederik Stork. A Computational Study on Bounding the Makespan Distribution in Stochastic Project Networks. *Annals of Operations Research*, 102(1–4):49–64, February 2001.

[11] Muthucumaru Maheswaran and Howard Jay Siegel. A Dynamic Matching and Scheduling Algorithm for Heterogeneous Computing Systems. In *7th IEEE Heterogeneous Computing Workshop (HCW'98)*, pages 57–69, Orlando, Florida, USA, March 1998.

[12] Hyunok Oh and Soonhoi Ha. A Static Scheduling Heuristic for Heterogeneous Processors. In Luc Bougé, Pierre Fraigniaud, Anne Mignotte, and Yves Robert, editors, *Proceedings of the Second International Euro-Par Conference on Parallel Processing-Volume II*, volume 1124 of *Lecture Notes in Computer Science*, pages 573–577, Lyon, France, August 1996. Springer.

[13] Rizos Sakellariou and Henan Zhao. A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems. In *13th IEEE Heterogeneous Computing Workshop (HCW'04)*, Santa-Fe, New Mexico, USA, April 2004.

[14] Vladimir Shestak, Jay Smith, Howard Jay Siegel, and Anthony A. Maciejewski. A Stochastic Approach to Measuring the Robustness of Resource Allocations in Distributed Systems. In *Proceedings of the 2006 International Conference on Parallel Processing (ICPP'06)*, pages 459–470, Columbus, Ohio, USA, August 2006.

[15] Zhiao Shi, Emmanuel Jeannot, and Jack J. Dongarra. Robust Task Scheduling in Non-Deterministic Heterogeneous Computing Systems. In *Proceedings of IEEE International Conference on Cluster Computing*, pages 1–10, Barcelona, Spain, September 2006. IEEE.

[16] Gilbert Sih and Edward Lee. A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogenous Processor Architectures. *IEEE Transactions on Parallel and Distributed Systems*, 4(2):175–187, February 1993.

[17] Haluk Topcuoglu, Salim Hariri, and Min-You Wu. Task scheduling algorithms for heterogeneous processors. In *8th IEEE Heterogeneous Computing Workshop (HCW'99)*, pages 3–14, San Juan, Puerto Rico, April 1999.